Project Report

on

Smart Recycling Contract

Mitha Rachel Jose

RDI-Developer

Digitalization Team

Centria

UNIVERSITY OF APPLIED SCIENCES

SMART RECYCLING CONTRACT

Background:

Plastic is among the most popular and important materials used in the modern world. However, its popularity is part of the huge problem and reason why plastics should be recycled. Instead of throwing them away polluting the land and our water bodies, we can optimize the lifespan of plastics by recycling and reusing them. Plastic recycling refers to the process of recovering waste or scrap plastic and reprocessing it into useful product. Due to the fact that plastic is non-biodegradable, it is essential that it is recycled as part of the global efforts to reducing plastic and other solid waste in the environment.

Recycling and recovering plastic are important in order to reach Finland's tightening recycling and climate objectives and to promote circular economy. The purpose of circular economy is to become less dependent on virgin raw materials and utilize existing materials more efficiently. The environment, climate and Finnish industry all benefit from more efficient recycling of plastic.

Advantages of Recycling Plastics

Plastics should be recycled because of a number of reasons as can be seen below:

1. Provision of a Sustainable Source of Raw Materials
2. Reduces Environmental Problems
3. Reduces Landfill Problems
4. Consumes Less Energy
5. Encourages a Sustainable Lifestyle among People

Objective:

To reduce the plastic waste in landfills and to create a sustainable environment to live using a recent technology called blockchain. With the help of human support and the technology we can build wonders with blockchain. The benefits of using the blockchain technology in smart recycling are

- Fraud and manipulation: When recyclers dispose the waste, they submit their waste plan report to the manufacture's for receiving the payment. Payments are based on how many kilograms waste has been disposed of. The smart recycling will examine the report and pay accordingly. Since

the reports can be easily manipulated, the payment frauds have been increased.

- Ensuring integral traceability: Every data entered on the blockchain is immutable and timestamped. Therefore, when it comes to paying the recyclers, the parties involved in the whole process can trace back the waste management activity history and can cross-check the submitted waste plan.

Proposal of Smart Recycling Contract

Smart Recycling Contract is a proposal for recycling the plastic materials generated by some companies and it should be recycled back to them. For instance, if a food packaging company selling their food packages and they are recycling the plastic waste of those food packages.

For this purpose, we made a smart decentralized application with a smart contract for the communication between the manufacture and the recycler. As the name indicates “SMART RECYCLING CONTRACTS”, it reduces the manpower needed to handle the recycling process. There is no intermediate person communicating between the manufacturer and the recycler.

The manufacturer can join into a block in the decentralized app using LOGIN and signed up as a manufacturer and he/she can able to add the products which they want to recycle by the recycler and send to the recycler.

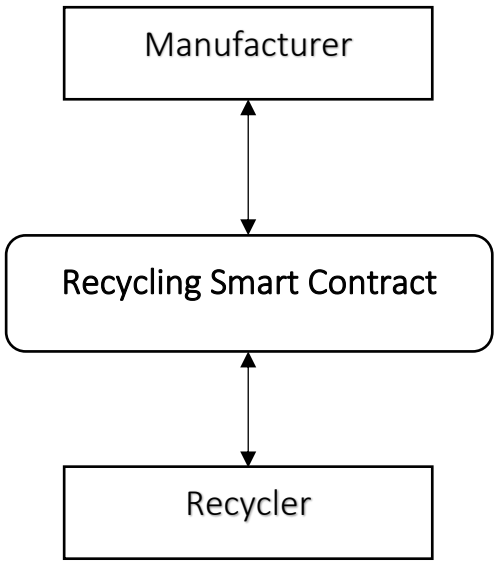
The recycler can also login to the decentralized app using a blockchain account and register or sign up as a recycler. Then the recycler can able to view the order from the manufacture and act based on the order. Once the recycler sorted the products of recycling, he will change the status that the ordered request from manufacture is recycled.

Once the status changed, the manufacture will get the recycled product and he should pay the recycler.

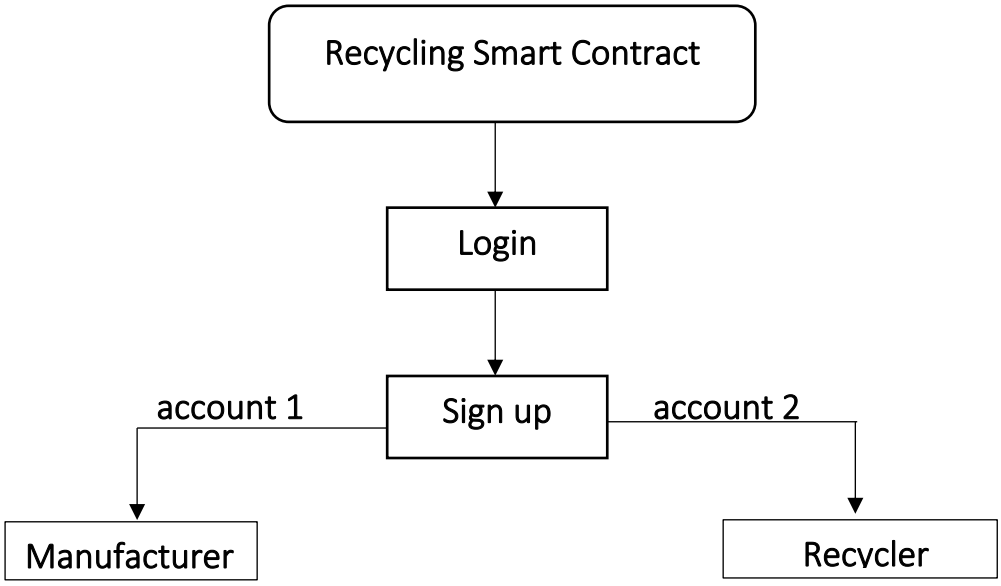
Data Flow Diagram of Smart Recycling Contract

The figure below shows a context Data Flow Diagram that is drawn for a Smart recycling Contract. It contains a process (shape) that represents the system to model. It also shows the participants who will interact with the contract, called the external entities and the entities who will interact with the system. In between the process and the external entities, there is data flow (connectors) that indicate the existence of information exchange between the entities and the system.

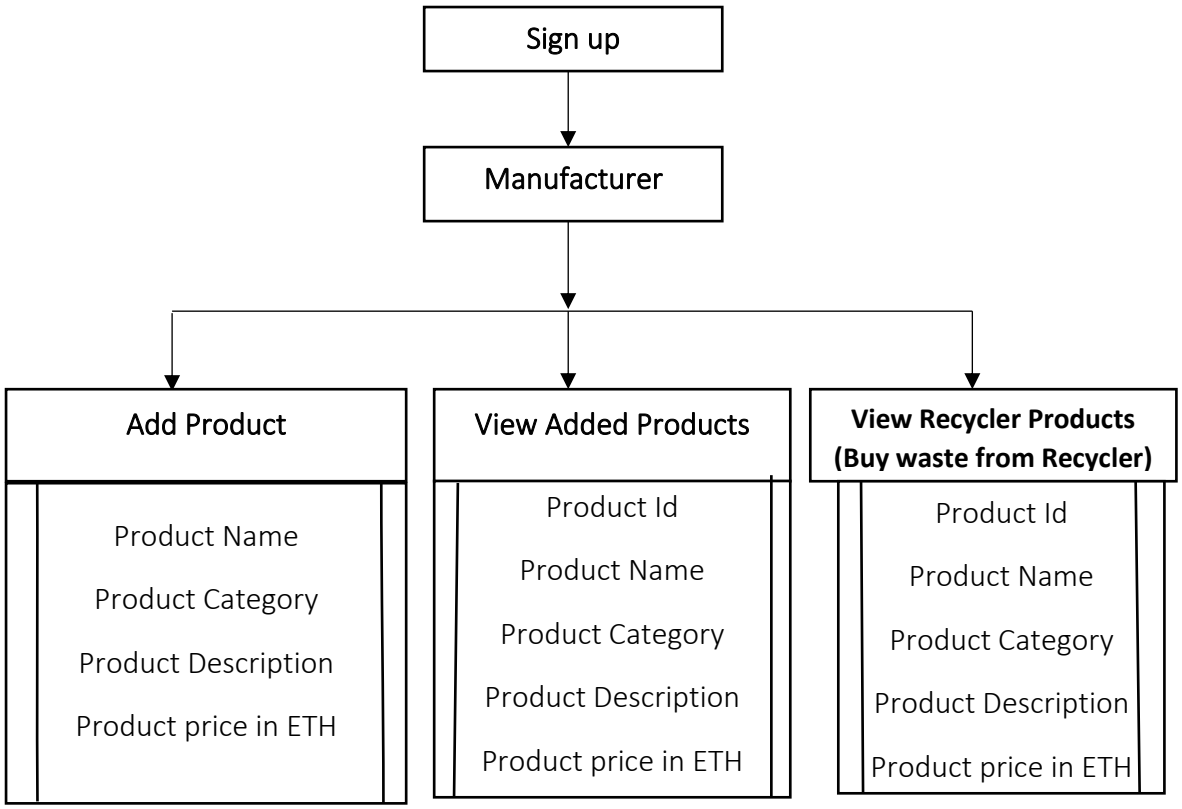
Level 0:



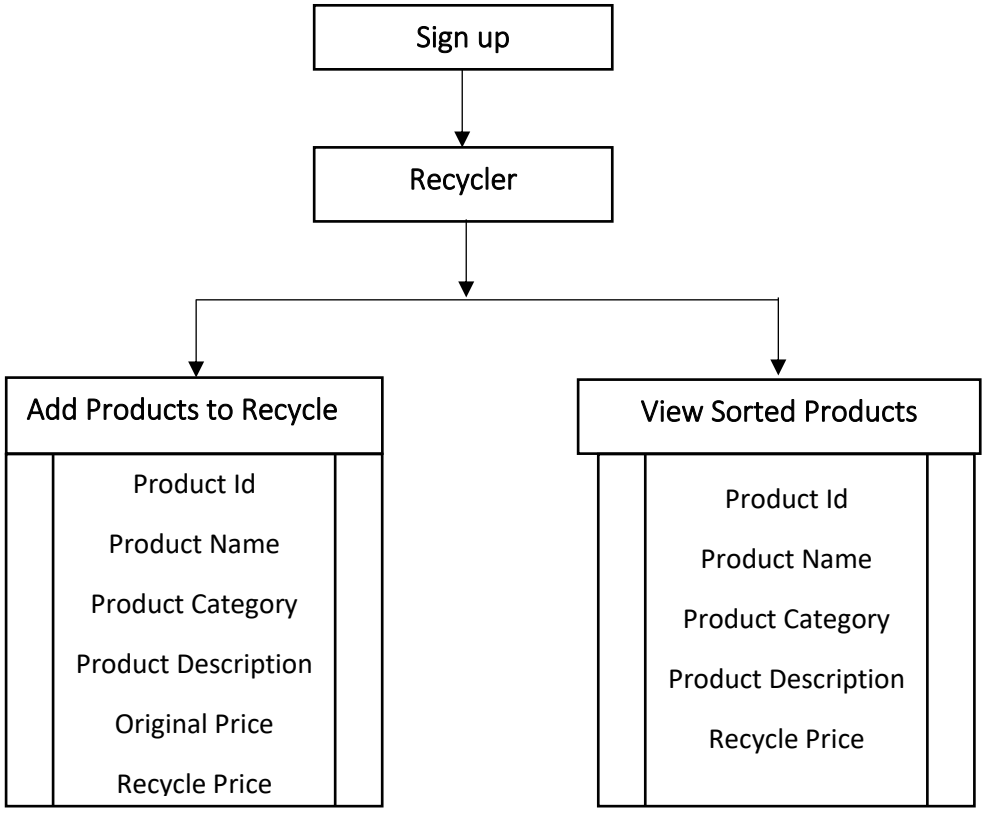
Level 1:



Level 2:



Level 3:



Software Requirements:

- Remix IDE: Remix is one of the easiest and browser-based tools to use for the creation and deployment of smart contracts.
- NPM
- Nodejs
- Truffle Framework: Truffle is a framework for Ethereum that offers a development environment for building Ethereum based apps.
- Solc: Solidity is a loosely typed programming language used for the creation of smart contracts on the Ethereum platform.
- Ganache: Ganache is a tool from Truffle Suite that allows developers to create their own private Ethereum blockchain to test dApps.
- Meta mask
- An editor- eg. Visual Studio code.
- Angular 6
- Programming Languages:

For smart contract: Solidity

Integration purposes: JavaScript, HTML, CSS, Ts, Angular cli or React ().

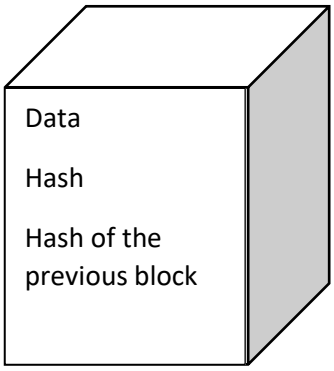
Description of Blockchain Technology:

A blockchain is a peer-to-peer network of computers, called nodes, that share all the data and the code in the network. Solidity is an object-oriented, high-level language for developing dApps (Decentralized applications), on the Ethereum blockchain. So, if you're a device connected to the blockchain, you are a node in the network, and you talk to all the other computer nodes in the network. In the Ethereum blockchain, miners work to earn Ether, a type of crypto token that fuels the network. Beyond a tradeable cryptocurrency, Ether is also used by application developers to pay for transaction fees and services on the Ethereum network.

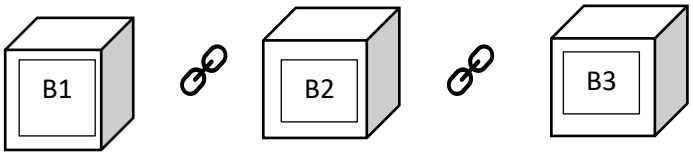
There is a second type of token that is used to pay miners fees for including transactions in their block, it is called gas, and every smart contract execution requires a certain amount of gas to be sent along with it to entice miners to put it in the blockchain.

Ethereum blockchain allows us to execute code with the Ethereum Virtual Machine (EVM) on the blockchain with something called a smart contract. Smart contracts are where all the business logic of our application lives – all variables and functions belong to a contract, and this will be the starting point of all your projects. Smart contacts are written in a programming language called Solidity, which looks like a mix of JavaScript and C.

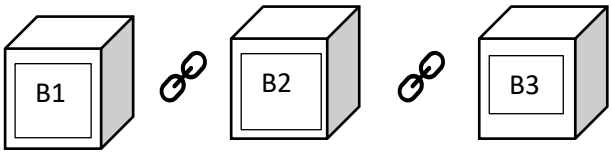
A blockchain is an "open distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way". A blockchain has a list of blocks. It starts with a single block, called the genesis block. Each of the block contains a data, hash (to uniquely identify the block) and hash value of the previous block.



Data: From address, To address, Amount (gas)
Hash: uniquely identify the block (alphanumeric value of size 12)
Hash of previous block



Hash:	1Z8F	6BQ1	3HI9
Previous hash value:	0000	1Z8F	6BQ1



Hash:	1Z8F	~~6BQ1~~ (H62Y)	3HI9
Previous value:	0000	1Z8F	6BQ1

If somebody tries to change the hash value of Block B2, then the previous hash value of Block3 is changed and the whole blockchain will be crashed.

A node is a device connected to the blockchain network that can initiate and verify transactions. Any device that wants to become a node needs to download a wallet. A wallet is a software that creates a pair of cryptographic keys for the node – a public key and a private key. A node's public key, which is accessible to everyone on the network, acts as its unique identifier for sending and receiving money. Blockchain wallets do not maintain a record of the owner's balance but have access to their copy of blockchain, which stores all transactions that ever happened on the network. So, the creator of each new transaction must specify all those transactions which brought him the money that he wants to send. Before a transaction appears as input, it is called Unspent Transaction Output (UTXO) and later it is referred to as a Spent Transaction. All UTXO specified as inputs within a transaction, must have the same receiver – the node that is creating the new transaction, and the received amounts must add up to at least the amount that the creator node is trying to send. To preserve the integrity of the newly created transaction, the creator node digitally signs it using its private key, which is hidden from the network. This transaction is then broadcasted to the neighboring nodes. When a peer receives a new transaction, it first verifies transaction integrity to ensure that it was actually signed by the creator node and was not tampered on its way. Integrity checks are followed by validity checks which verify that the inputs are UTXO, they were sent to the creator node and the total of all inputs is at least equal to the amount being transferred by the creator node. Each node uses its copy of the blockchain to track the origin of the specified UTXO.

Once validated, the node adds this transaction to its MemPool and relays this transaction to its neighboring nodes. These transactions are picked up by Miners and grouped together in a block. Miners are nodes that take the responsibility of appending new blocks to the blockchain. Each block contains a block number, a timestamp, hash of the previous block and a subset of transactions. A block's hash is a unique 64-digit hexadecimal number, generated by feeding the block's contents into SHA256 hashing algorithm. SHA256 converts input of any length and type, into a unique 64-digit hash. This hash changes completely even with a single character modification in the input. Fig.2 displays the presence of previous block's hash in each subsequent block links all the blocks together as a blockchain. Adding a new block requires computing the hash of the new block. This task is made highly difficult by restricting the value of the calculated hash to

be lower than a specific threshold value. Since the block contents are static and will always result in the same hash, a variable value called Nonce is included in block contents to find a hash below the target. Miners across the network race against each other to find an appropriate nonce for the next new block. The first one to find a value which makes the hash of the next block below the specified target gets newly created bitcoins as a reward. This process is called Mining. Once a miner mines the new block, it is relayed over the network to be confirmed by other nodes and miners. When majority of peers agree on the correctness of the new block (Consensus), it is appended in the blockchain. Modifying the blocks which are a part of the blockchain, requires huge computational power and speed. If a malicious node tries to alter a block, it would change the hash of the altered block and would not match the hash present in next block (in the field: Previous Block Hash). Due to invalid hash in the contents, the next block's hash also becomes invalid. All the subsequent blocks become invalid.

Security Mechanisms Used in Block Chain:

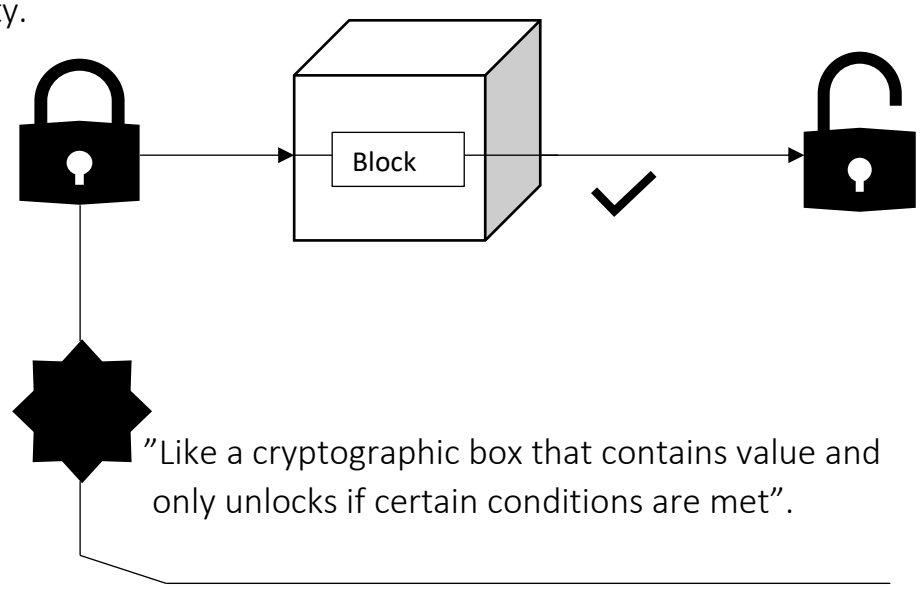
- Consensus Algorithm
- P2P Network

Consensus Algorithm: This is also called Proof of Work. Basically, this is a cryptographic puzzle that the computer has to solve whenever wants to puta new block on the chain. It takes at least 10 minutes time to solve it. This process is called mining.

P2P Network (peer to peer Network): Everyone who wants to join on the network can enter into the network. There are lots of blocks on the network and if a new block added to the network it should be verified and that block can also able to see all transactions. If someone tries to cheat the system and all in the chain is not agreed with the new block, then we can understand that a hacker is on and neglect that connection or block.

Smart Contracts:

A smart contract is a self-enforcing agreement embedded in computer code managed by a blockchain. The code contains a set of rules under which the parties of that smart contract agree to interact with each other. Smart contracts help you exchange money, property, shares, or anything of value in a transparent, conflict-free way while avoiding the services of a middleman. Contracts can be encoded on any blockchain, but Ethereum is mostly used since it gives unlimited processing capability.



Ethereum is a platform that allows its users to create decentralized applications (Dapps) using blockchain technology. Ethereum is not only useful for the execution of normal transactions using the Ether cryptocurrency, but it also facilitates the creation of programs on blockchain using a Turing complete language called Solidity. These codes, which are executed on the Ethereum platform, are called Smart Contracts. Nodes can create and deploy smart contracts on the blockchain to act as an agreement between users. Each node in the Ethereum blockchain network maintains the current state of the smart contract and monitors the actions performed by it. Ethereum Virtual Machine executes the smart contract code when required data are provided to it by a node or other smart contract. Smart contracts not only define the rules and penalties of the agreement, just like a normal contract, but also enforce these whenever required. This is done without the need for external enforcement parties. Using smart contracts, rules can be implemented in an easy and hassle-free manner. Any two parties, who decide to abide by a given set of rules, can use these contracts. Ethereum allows such parties to code all the regulations, the penalties and incentives, into a smart contract. If any of these parties do not follow the contract rules, necessary actions, as defined in the code, are undertaken. Hence

interacting parties need not trust each other or a third party in case of an agreement based on smart contracts.

Benefits of smart contract:

- Immutable
- Transparent
- Distributed

Sample Code of Smart Contract:

```
pragma solidity ^0.5.16;

contract Ewaste{

// Variables

uint public U_ID = 1000;

uint public P_ID = 1;


// Structs

struct Products{

  string productName;

  string category;

  string description;

  uint manufactureId;

  uint price;

  uint recyclerID;

  bool status;

}

struct Users{

 address payable userAddress;

 string userName;

 string contactAddress;

 uint accountType;

//  uint MyProducts;

}

// Mappings
```

```
mapping(uint => Products)public Product;

mapping(uint => Users) public User;

mapping (address => uint) public UserID;


//Modifiers


modifier onlyMan( uint mID) {

  require(User[mID].userAddress == msg.sender && User[mID].accountType == 1,'only
Manufacture');

  _;

}

modifier onlyRec(uint rID) {

  require(User[rID].userAddress == msg.sender && User[rID].accountType == 2,'only Recycler');

  _;

}

modifier recycler(uint pID) {

  require(Product[pID].recyclerID == 0 && Product[pID].status == true,'Recycler Already Asigned ');

  _;

}

modifier buyProduct(uint pID,uint mID) {

  require(Product[pID].manufactureId == mID,'its Not your Product');

  require(Product[pID].status == false,'Not a Waste');

  _;

}

//Functons


function Registration (string memory userName,string memory contactAddress,uint accountType)
public{

    require(UserID[msg.sender]==0,'User Already Registred');

    require(accountType == 1 || accountType == 2 , ' Invalid accountType');

  User[U_ID].userAddress = msg.sender;

  User[U_ID].userName = userName;
```

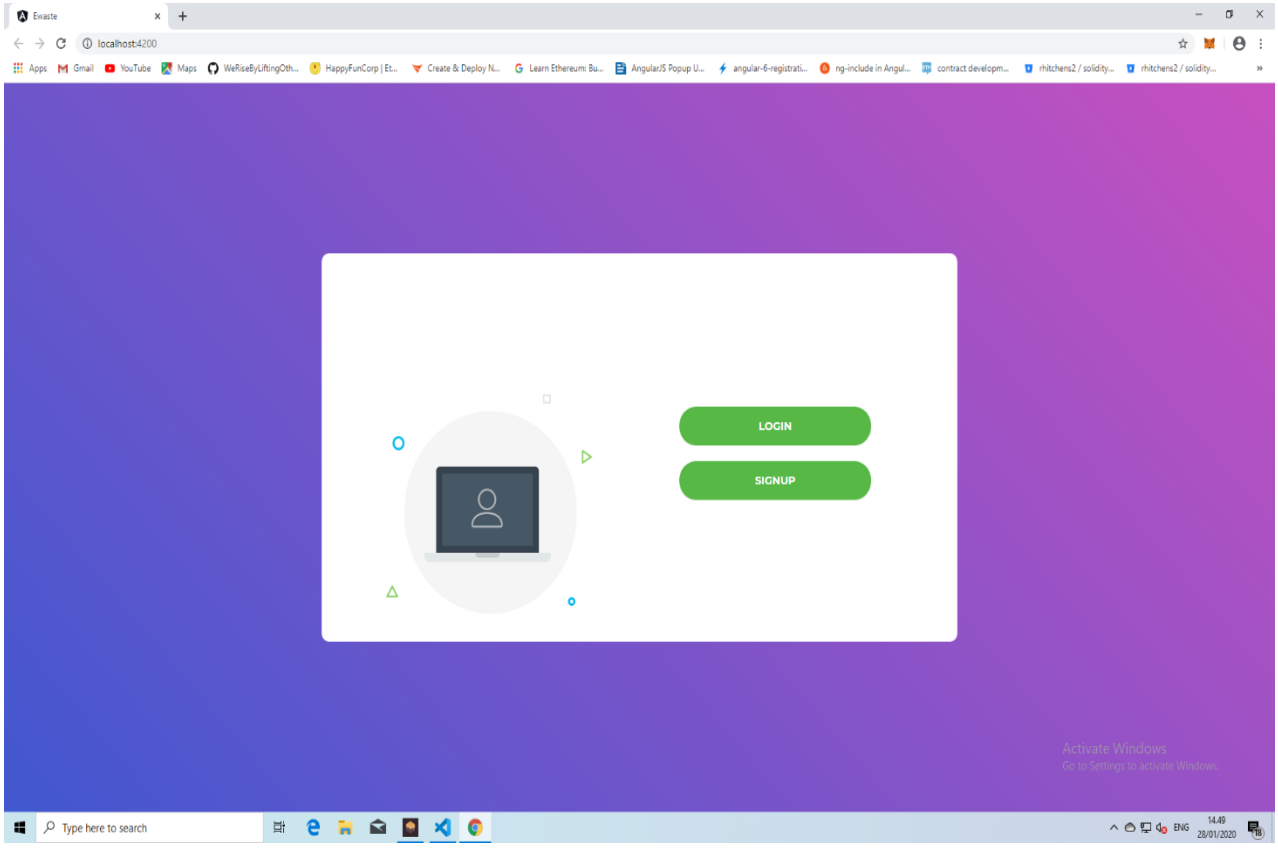
```
  User[U_ID].contactAddress = contactAddress;
  User[U_ID].accountType = accountType;
  UserID[msg.sender] = U_ID;
  U_ID++;
}
function CreateProduct (uint _M_ID ,string calldata _name,string calldata _category,string calldata
_description,uint _price) external onlyMan(_M_ID) {
  Product[P_ID].manufactureId = _M_ID;
  Product[P_ID].productName = _name;
  Product[P_ID].category = _category;
  Product[P_ID].description = _description;
  Product[P_ID].price = _price;
  Product[P_ID].status = true;
  P_ID++;
}
function SortRecycler(uint _R_ID,uint _P_ID) public onlyRec(_R_ID) recycler(_P_ID)  {
  Product[_P_ID].recyclerID = _R_ID;
  Product[_P_ID].status = false;
}
function BuyProduct(uint _P_ID,uint _M_ID) public payable onlyMan(_M_ID)
buyProduct(_P_ID,_M_ID){
  require(Product[_P_ID].price == msg.value,'Incorret Price');
  User[Product[_P_ID].recyclerID].userAddress.transfer(Product[_P_ID].price);
  Product[_P_ID].recyclerID = 0;
  Product[_P_ID].status = true;
}
}
```

Steps to follow:

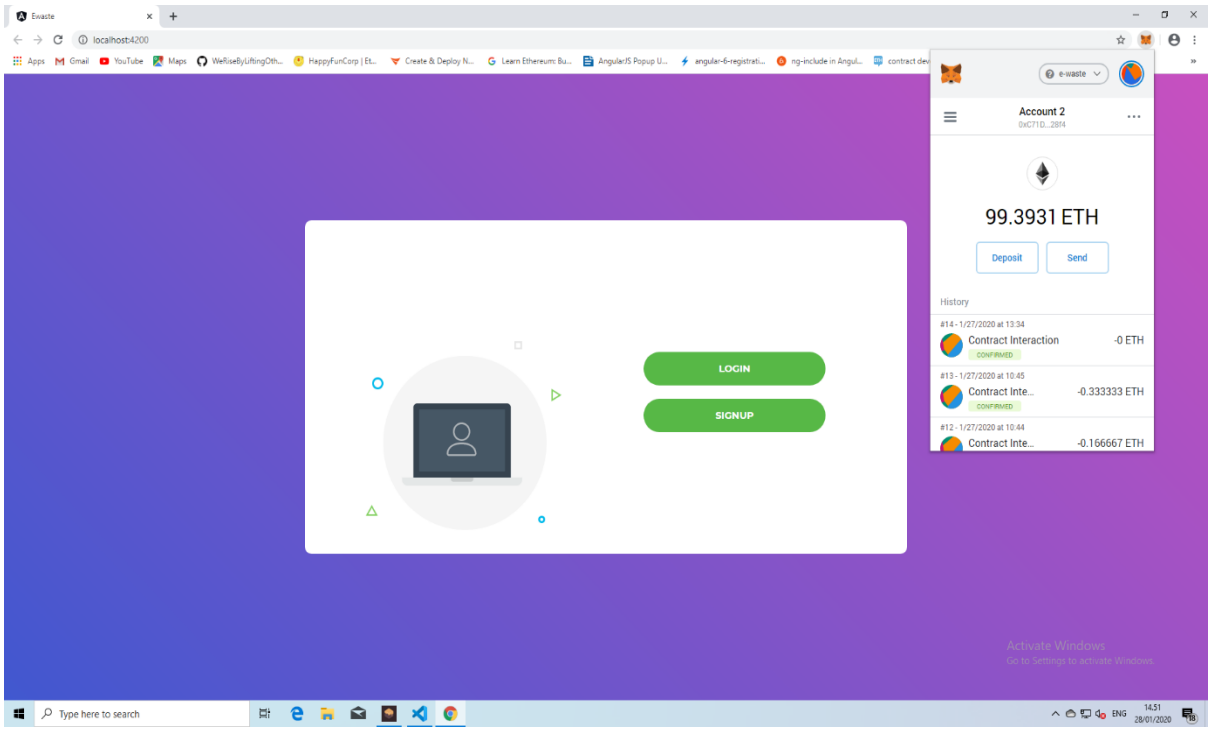
Step 1: Open Visual Studio Code – Add project from the folder
Step 2: Open Ganache in Quick Start
Step 3: Open Meta mask in browser – Import 3 accounts using Ganache
Step 4: Back to Visual studio code and run in the terminal – truffle migrate, ng serve.
Step 5: Open in browser localhost:4200. Then the dApp will open
Step 6: Click Login. Then the meta mask open by itself asking to connect to the contract.
Step 7: Sign Up as manufacturer and add the product details.
Step 8: Log out
Step 9: Login as recycler and the product to be recycled.
Step10: Log out

Screenshots of Smart Recycling Contract:

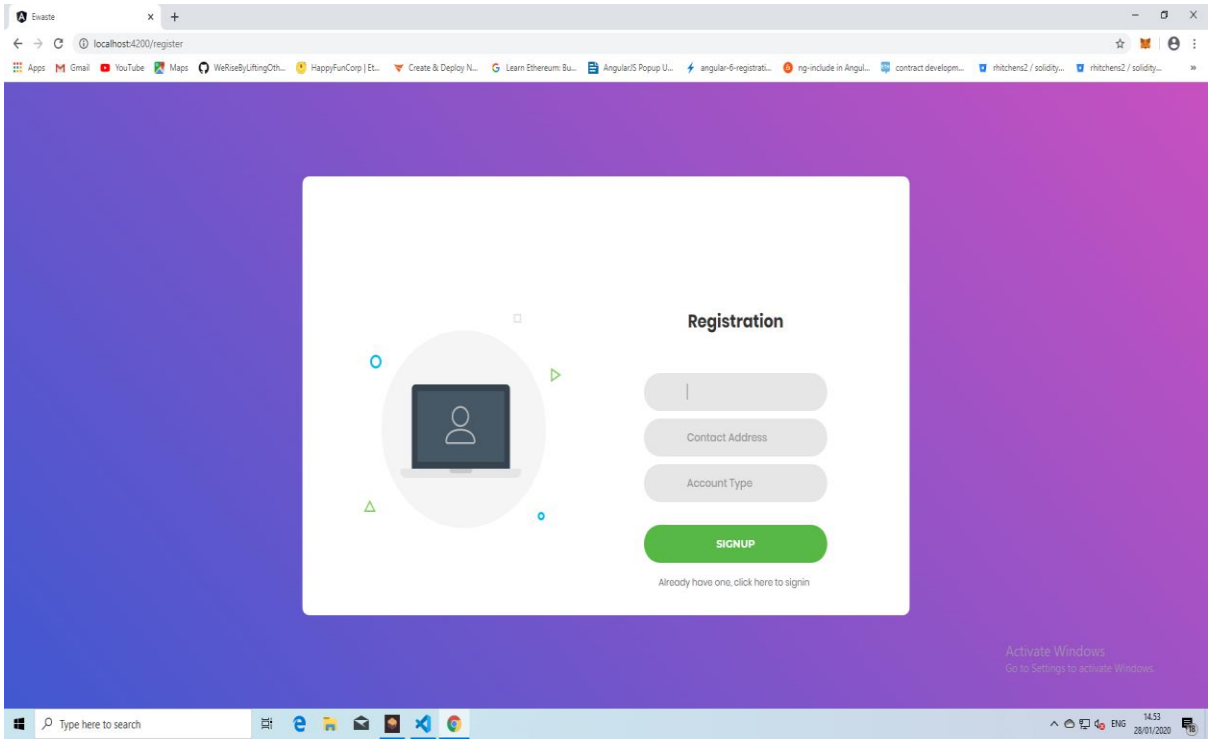
Login Page



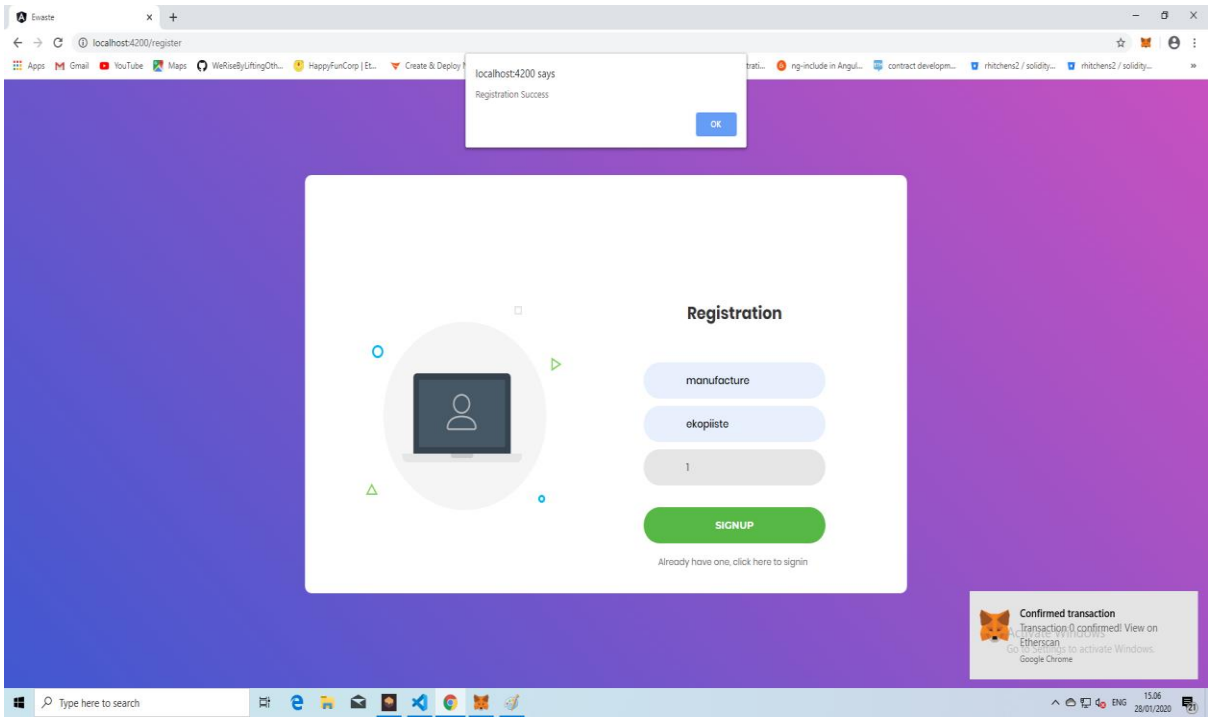
Login with Meta mask



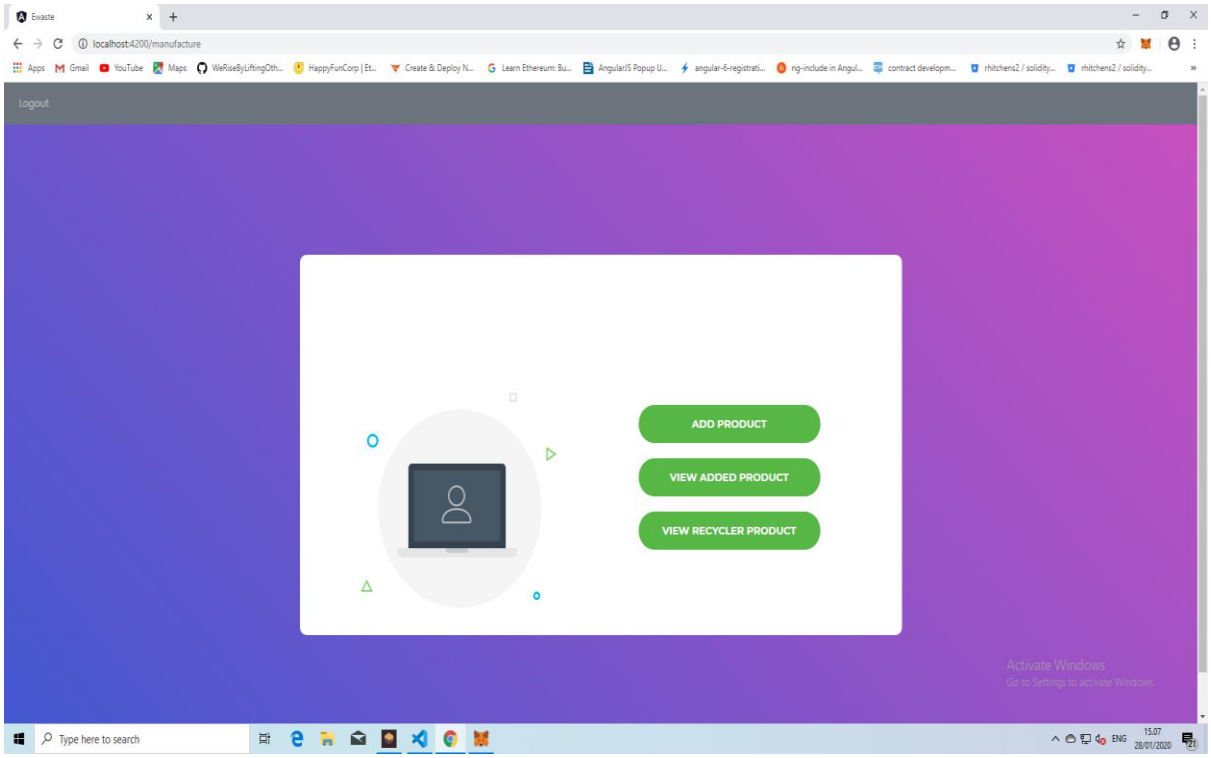
Registration field



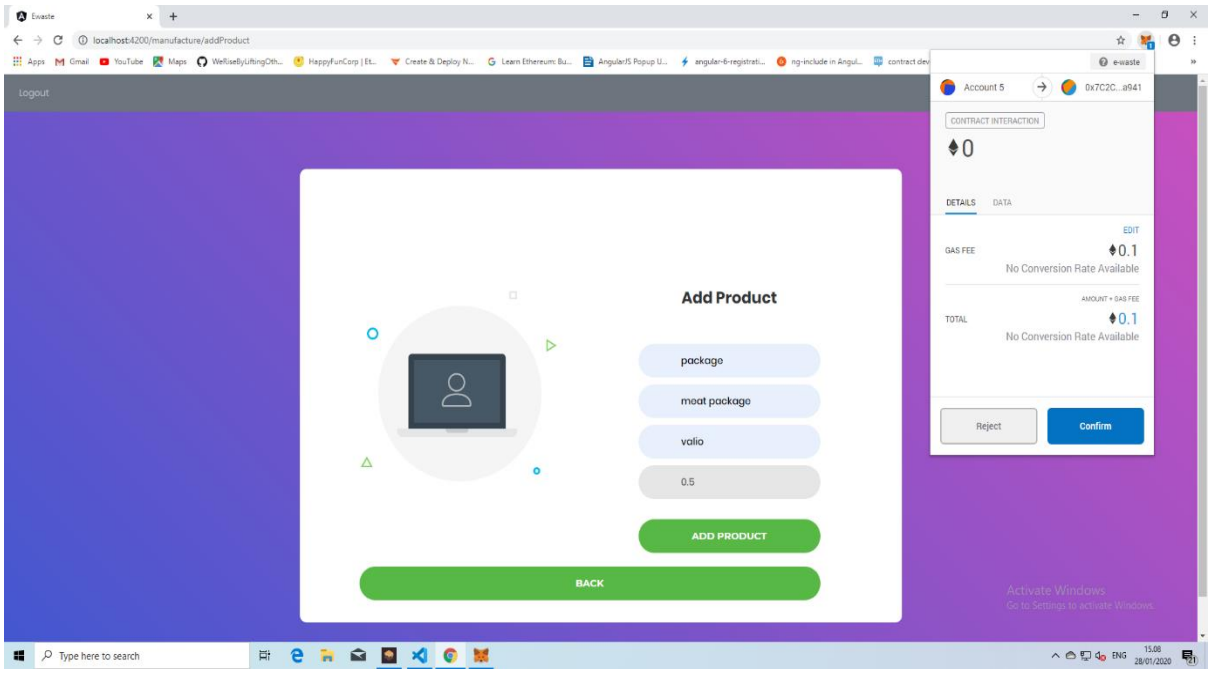
Registration Success



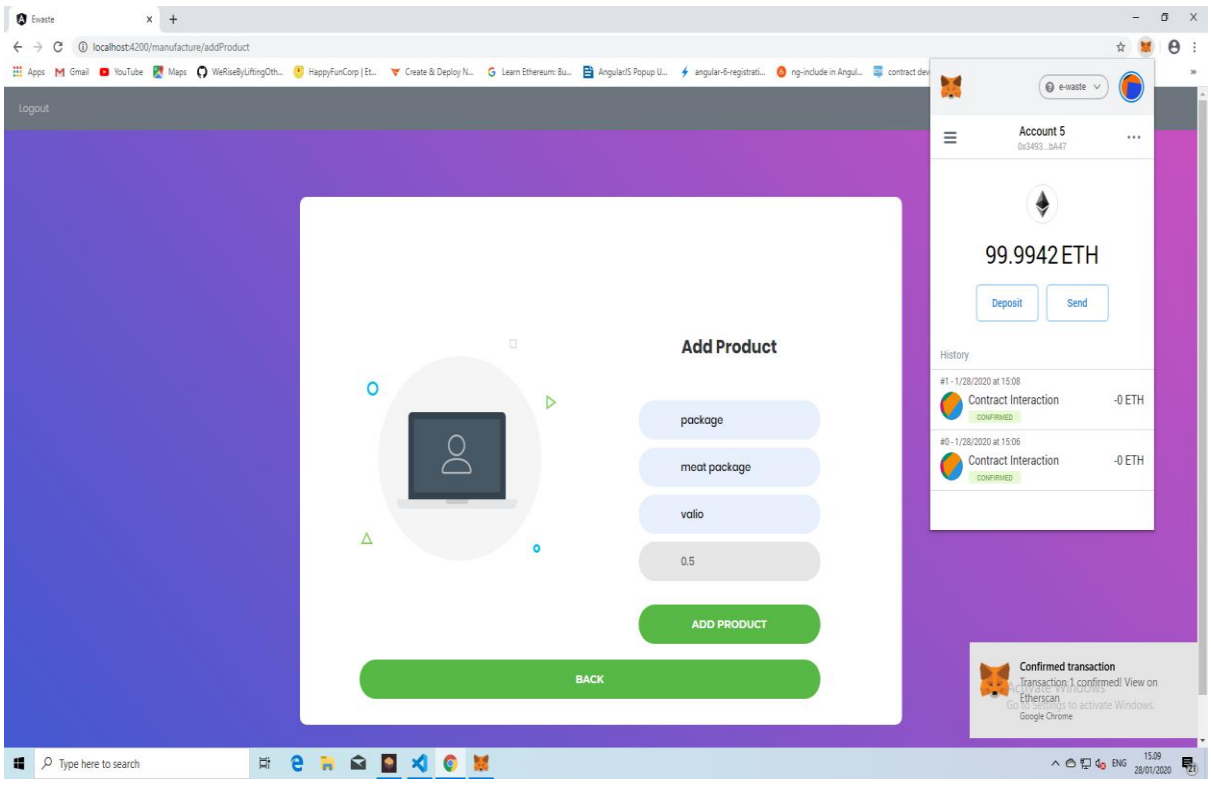
After Registration



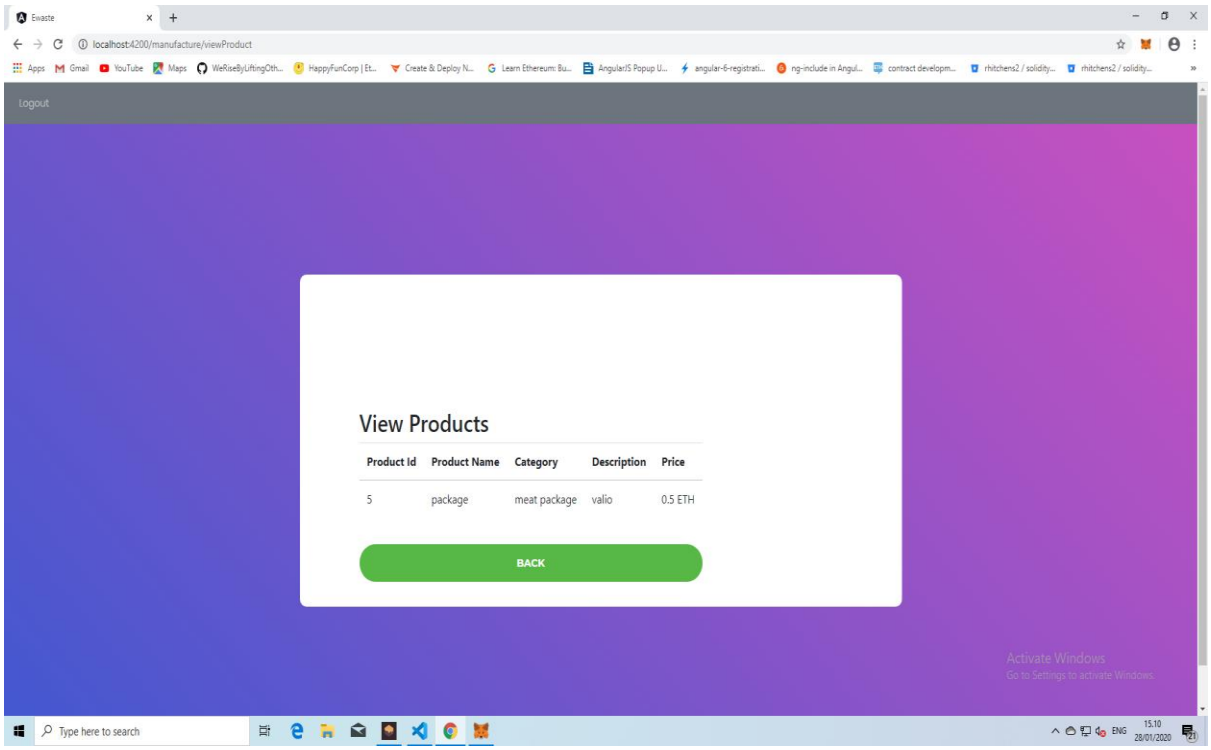
Adding the Product by the Manufacturer



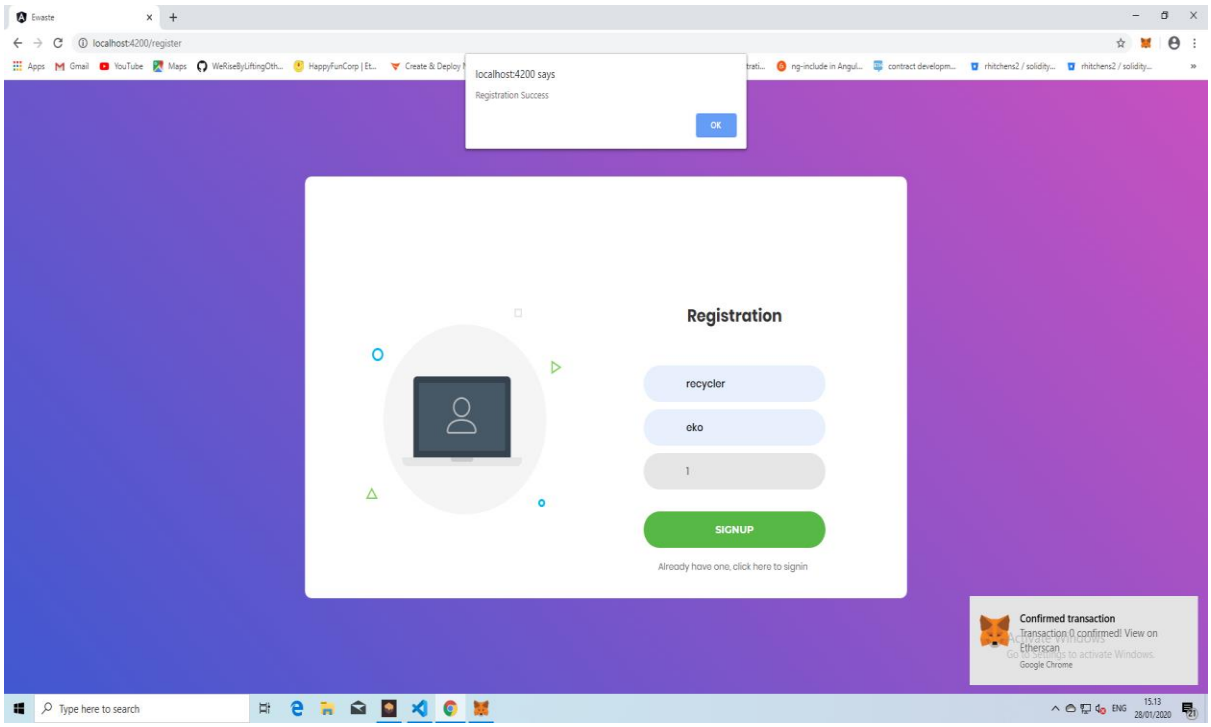
Confirmed the transaction for adding the Product



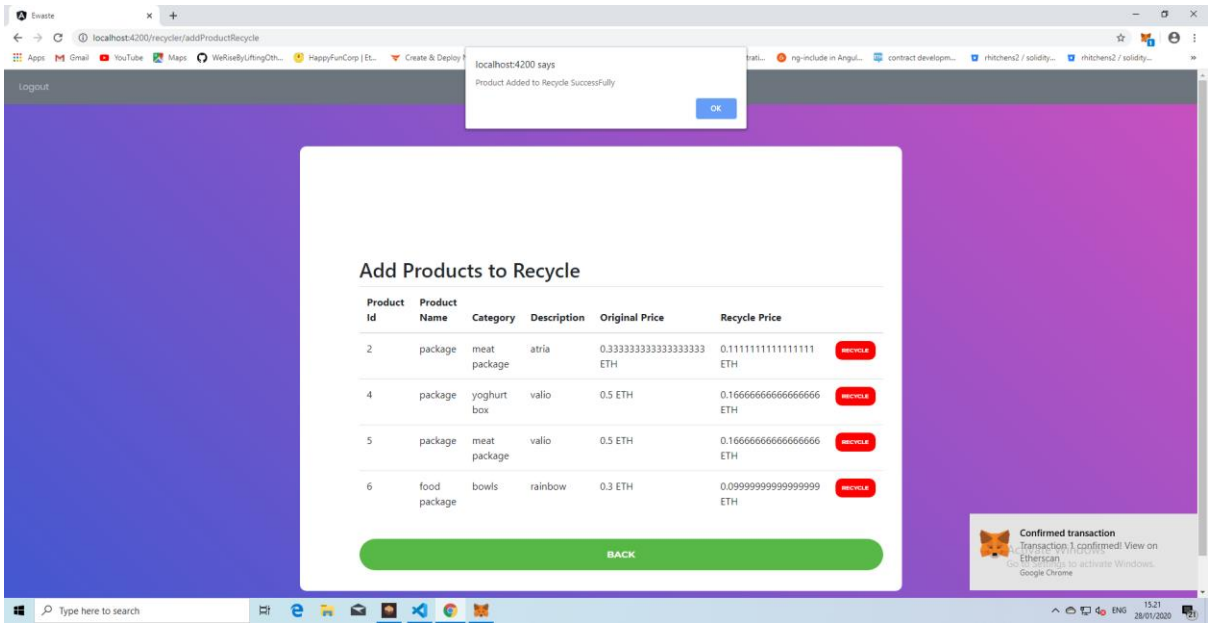
Products saved in the view list



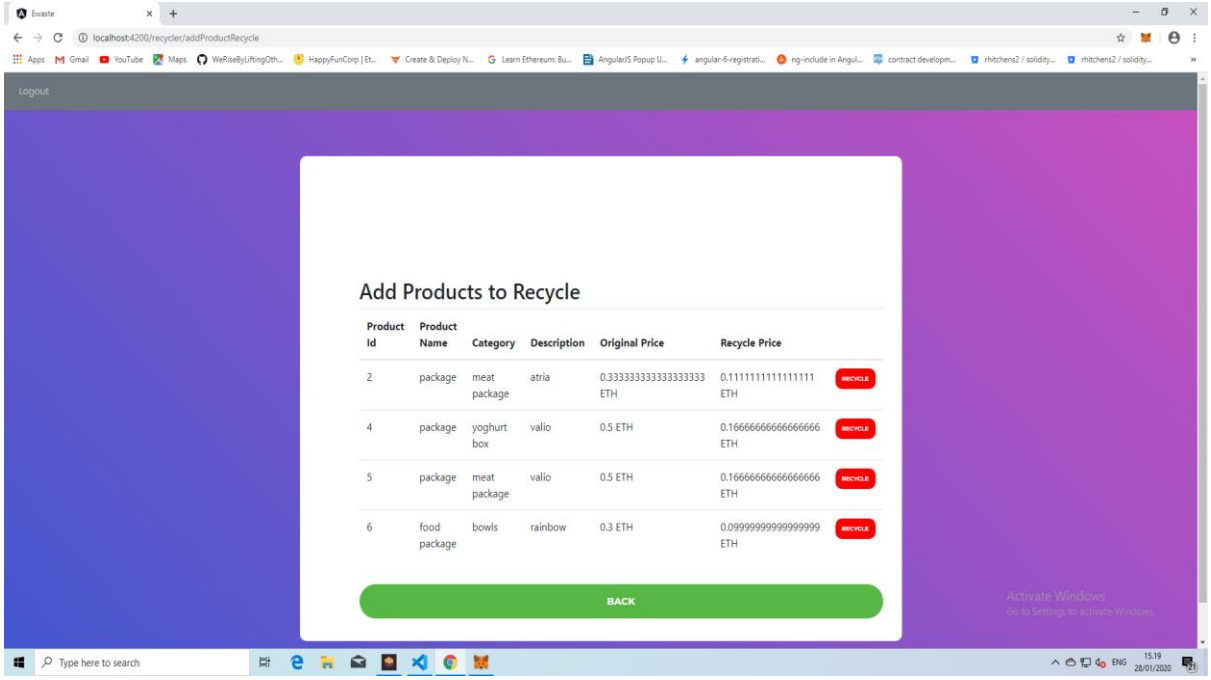
Registration Successful for Recycler



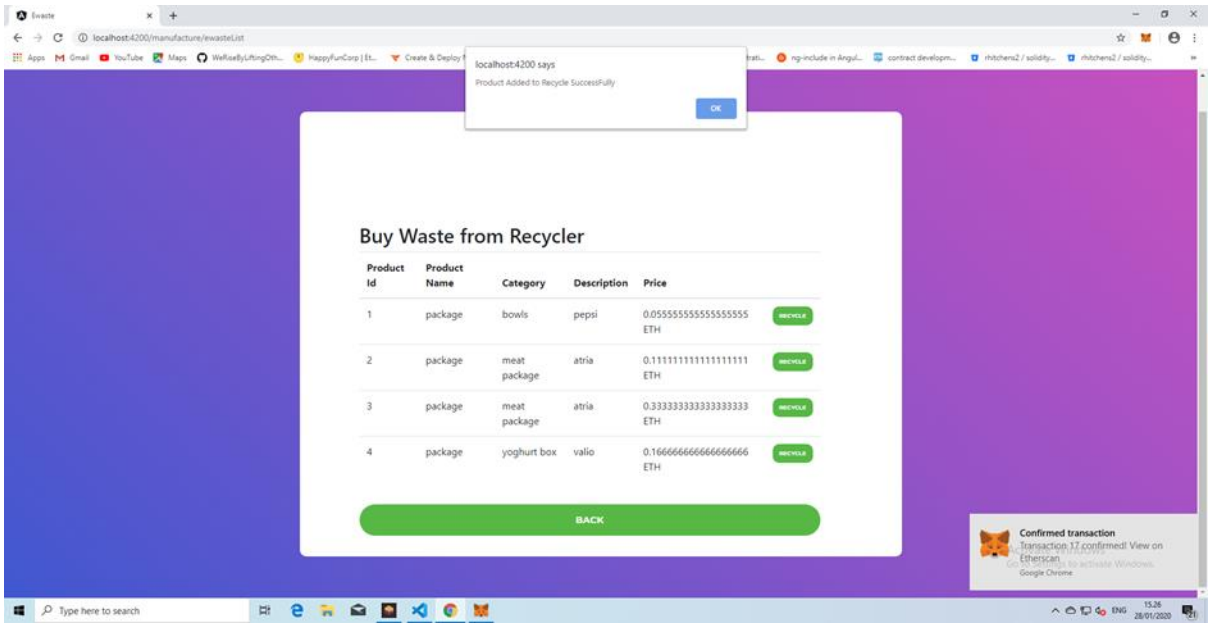
Products Added to Recycle



Product added successfully to recycle



Product adding for Recycling transaction Confirmed



Conclusion:

Blockchain-based smart contract has become a growing field in the blockchain technology. This project tries to implement a small blockchain contract for recycling with a user interface. The security of a smart contract needs to be built properly in order for it to be considered safe to use. Smart contracts have certain advantages for many industry sectors such as reducing overhead costs, providing transparency and saving time. They are more reliable, secure, efficient and trustworthy as compared to paper contracts.

Future Plans

Waste has always been difficult to track but with blockchain, it is believed that it would be much easier. It wouldn't be so difficult to design a system where pieces of plastic (or trash bags) could be tagged with scannable QR-Codes and then tracked at each step of the recycling chain.

Currently, waste management companies and also recycling companies are working on improving their techniques and tools. There is a recent development available in single-stream recycling which allows people to dump all their trash in one bin. That has helped to reduce people's burden of sorting and has also impacted the recycling rate.

An innovative device (Automated recycling bin) that can autonomously sustain the ecosystem and retaining its core fundamental working principles. This device has some features which are not present in traditional waste management systems. Some of the features include: Dapp Support, IoT support, Automatic opening and closing lid, LED Panels, sorting waste at source, compressing waste, a unique User Identification.

References

1. Ishan Mistry, Sudeep Tanwar, Sudhanshu Tyagi, Neeraj Kumar," Blockchain for 5G-enabled IoT for industrial automation: A systematic review, solutions, and challenges", Mechanical Systems and Signal Processing 135 (2020) 106382, 11 October 2019.
2. Xiaonan Wang, Wentao Yang, Sana Noor, Chang Chen, Miao Guo, Koen H. van Dam," Blockchain-based smart contract for energy demand management" 10th International Conference on Applied Energy (ICAE2018), Hong Kong, China, 22-25 August 2018.
3. Daniel Macrinici, Cristian Cartofeanu, Shang Gao," Smart contract applications within blockchain technology: A systematic mapping study", Telematics and Informatics 35 (2018) 0736-5853, https://doi.org/10.1016/j.tele.2018.10.004.
4. A.S.L. França, J. Amato Neto, R.F. Gonçalves, C.M.V.B. Almeida, Proposing the use of blockchain to improve the solid waste management in small municipalities, Journal of Cleaner Production (2019), https://doi.org/10.1016/j.jclepro.2019.
5. Alberto Attilio Brincat, Alfio Lombardo , Giacomo Morabito , Salvatore Quattropanic " On the use of Blockchain technologies in WiFi networks",07.011 1389-1286, https://doi.org/10.1016/j.comnet.2019.
6. Bhabendu Kumar Mohanta, Debasish Jena, Soumyashree S. Panda, Srichandan Sobhanayak, "Blockchain technology: A survey on applications and security privacy Challenges", 2542-6605, https://doi.org/10.1016/j.iot.2019.100107 .
7. Neha Gupta, Punam Bedi ,"E-waste Management Using Blockchain based Smart Contracts", Conference Paper, September 2018, DOI: 10.1109/ICACCI.2018.8554912.
8. Six Control Principles for Financial Services Blockchains, October 2017, This publication, prepared during the summer months of 2017 by the Deloitte EMEA Blockchain Lab in Dublin in association with Deloitte Hong Kong and US, explores six control principles essential for blockchain adoption on a global scale.
9. https://www.italtel.com/focus-how-to-use-the-blockchain-for-better-cybersecurity/
10. https://www.forbes.com/sites/andrewarnold/2019/01/30/4-promising-use-cases-of-blockchain-in-cybersecurity/#36b037c3ac32
11. https://esatya.io/blood-chain-tracking-blood-journey-through-blockchain/
12. https://consensys.net/blockchain-use-cases/digital-identity/

13. https://www.udemy.com/course/understanding-blockchain technology/learn/lecture/9452500#questions
14. https://www.dappuniversity.com/articles/solidity-tutorial
15. https://medium.com/coinmonks/creating-smart-contracts-with-smart-contract-d54e21d26e00
16. https://hackernoon.com/smart-waste-management-and-blockchain-technology-887a8a185357
17. https://medium.com/crypto-currently/build-your-first-smart-contract-fc36a8ff50ca